Escuela Politécnica Superior

Recopilación de información sobre fútbol y predicción de resultados

Grado en Ingeniería Informática

Trabajo Fin de Grado

Autor:
Adrián Bardisa Serrano

Tutor/es:
Juan Ramón Rico Juan
A. Javier Gallego Sánchez

Junio 2019

Universitat d'Alacant
Universidad de Alicante

Recopilación de información sobre fútbol y predicción de resultados

Estudio de la Liga Española, recopilación de datos y predicción usando modelos estadísticos y de aprendizaje automático

Autor
Adrián Bardisa Serrano

Directores
Juan Ramón Rico Juan
Departamento de Lenguajes y Sistemas Informáticos
Antonio Javier Gallego Sánchez
Departamento de Lenguajes y Sistemas Informáticos

Grado en Ingeniería Informática

Escuela Politécnica Superior

ALICANTE, 27 de mayo de 2019

Justificación y Objetivos

El principal objetivo del trabajo es recopilar información sobre los partidos de La Liga Española y de sus jugadores. Se pretende diseñar e implementar una base de datos que contenga toda la información estructurada y sin duplicidades, con el fin de poder obtener la información deseada mediante consultas a esta base de datos.

También queremos aplicar un modelo estadístico para intentar predecir que equipo va a ganar en un encuentro y compararlo con algún modelo de aprendizaje automático basado en Deep Learning.

Agradecimientos

En primer lugar, me gustaría agradecer el trabajo a mis dos tutores, tanto a Juan Ramón como a Javier por la ayuda y libertad que me han proporcionado para llevar a cabo el trabajo.

También quiero dar las gracias a mi familia por apoyarme en todo el trayecto.

Por último, quisiera agradecer a un amigo apodado *micki* por todas las charlas que le he dado contándole el trabajo y aún sin entender nada seguir escuchándome hasta el final.

Si consigo ver más lejos
es porque he conseguido auparme
a hombros de gigantes

Isaac Newton.

Índice general

Índice de figuras

Índice de tablas

1 Introducción

Todos sabemos que el deporte forma una gran parte de nuestra sociedad. Actualmente el término deporte no sólo abarca las actividades que necesariamente tienen que implicar un ejercicio físico, existen deportes electrónicos, también conocidos como E-Sports. Debido a todos los nuevos deportes que han nacido, han salido nuevos modelos de negocio como las casas de apuestas que básicamente te proponen una cuota (un multiplicador) para alguna característica del evento, es decir, apuestas dinero y si ganas te devuelven el dinero apostado multiplicado por la cuota.

En este TFG nos vamos a centrar en el deporte de fútbol (también conocido como *football soccer*) en la liga española y en intentar predecir el resultado del encuentro entre dos equipos.

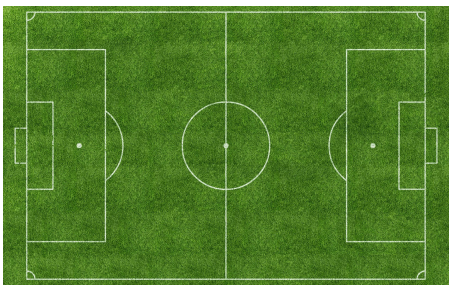

Figura 1.1: Campo de fútbol

El fútbol es un deporte en el que se enfrentan dos equipos de 11 jugadores cada uno en un campo como el de la imagen 1.1, uno de ellos es el portero que es el que se encarga de proteger la portería de su equipo, tiene un área en la que puede coger la pelota con las manos. Los demás jugadores se organizan en formaciones que pueden ser más ofensivas o defensivas. En esas formaciones existen 3 posiciones básicas, los defensas que juegan más cerca de la portería, los centrocampistas que juegan por el centro del campo y los delanteros que su tarea principal sería la de anotar goles en la portería del equipo contrario.

El juego consiste en mover una pelota por el campo delimitado por líneas y intentar meter la pelota en la portería del equipo contrario, a esto se le conoce como gol. En un enfrentamiento entre dos equipos puede ganar el que juega como local si mete más

goles que el equipo visitante, puede quedar empate si anotan los mismos goles o ganar el visitante si este mete más goles que el local.

La competición principal en cada país es la Liga (en cada pais se llama de una forma distinta). Se disputa entre 20 equipos distintos y cada equipo juega contra los otros 2 veces, una vez en el campo local y otra en el campo del otro equipo. Después de cada partido se obtienen 3 puntos si ganas, independientemente de si estas jugando como local o como visitante, y 1 si empatas. Las jornadas constan de 2 ó 3 días en los que se disputan los partidos entre los equipos, jugando cada equipo una vez como máximo. Al final de todas las jornadas, al terminar la Liga, gana el que mayor puntuación haya obtenido y, en caso de empate, se utilizarían los goles para desempatar, adquiriendo una mejor posición el que mayor número de goles haya encajado en toda la Liga.

Más del 40 % de la gente se considera fan del fútbol 1.2 , esto hace que sea el deporte más famoso en todo el mundo [1]. Especialmente en Europa, desde términos de los ingresos y el nivel de las competiciones, es el mercado más grande de fútbol en el mundo. Las imagen 1.3 muestra como han ido aumentando los ingresos durante los años [2] y aún se espera que sigan aumentando. Debido a este gran aumento de ingresos, han nacido una gran cantidad de casas de apuestas y gente conocida como *tipster*.

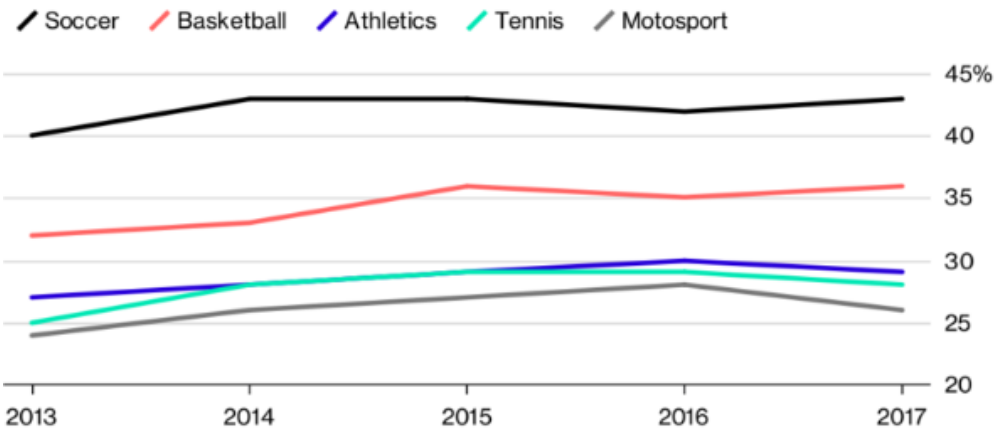

Figura 1.2: Porcentaje de personas que dicen que están muy interesadas o interesadas en cada deporte por América, Europa, el medio Oriente y Asia.

Tipster o analista es una persona que se dedica a estudiar los resultados en un deporte, cómo influyen los diferentes factores en esos resultados y que intenta predecir el resultado del siguiente encuentro. Su modelo de negocio consiste en publicar esas apuestas en un grupo que necesita suscripción y además apostar en esa misma apuesta que creen que es fiable.

Hace tiempo sólo se podía intentar pronosticar la quiniela, cosa que es muy difícil, tienes muy pocas probabilidades de ganar, ya que hay que acertar casi todos los partidos

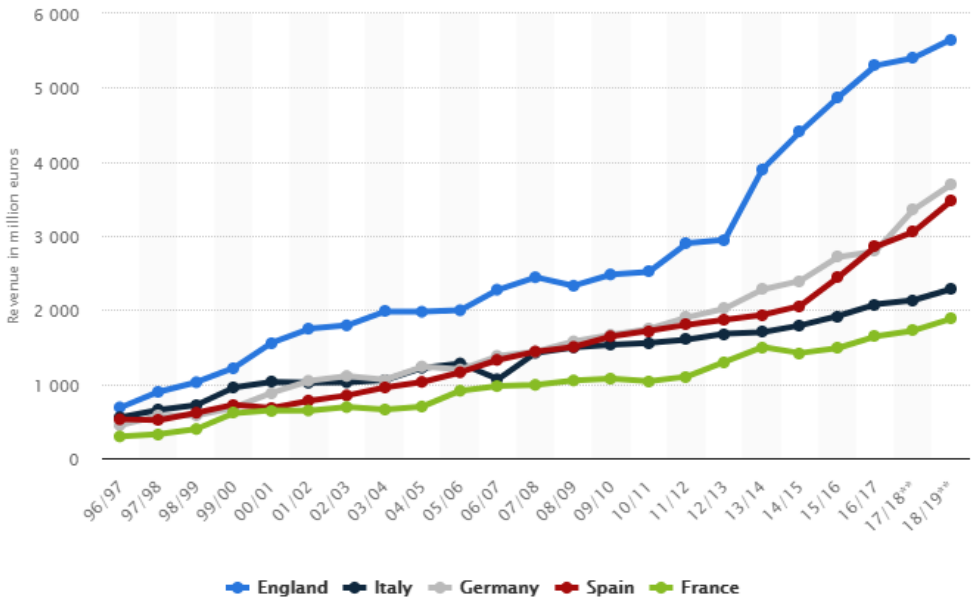

Figura 1.3: Ingresos (en millones de euros) de las 5 competiciones más importantes de fútbol desde 1996/97 hasta 2018/19.

y estadísticamente la probabilidad es muy baja. Actualmente, con todo el aumento de casas de apuestas, se puede pronosticar un campo muy amplio de resultados en el partido, desde los goles hasta cosas como el número de faltas que se realizan en el encuentro.

Desde que aparecieron, los ordenadores se han encargado de realizar de forma más rápida que nosotros muchas tareas tediosas y repetitivas. Un ordenador es capaz de calcular muchísimo más rápido que un humano, sin embargo, hay ciertas tareas que para nosotros son triviales y que ellos no han sido capaces de realizar con tanto éxito. Me refiero a problemas como la visión, el reconocimiento del sonido o el autoaprendizaje. Los humanos aprendemos a reconocer formas y sonidos desde una edad muy temprana, pero para los ordenadores les supone un gran desafío. Es precisamente en ese tipo de tareas donde se aplica el aprendizaje automático (en inglés *Machine Learning*, ML). Los algoritmos de ML aprenden a realizar tareas generalizando a partir de los ejemplos aportados. Es una buena forma de resolver problemas que no se pueden resolver matemáticamente, es decir, aplicando una función o ecuación conocida.

Por ello, vamos a usar ML para intentar resolver el problema que se nos plantea. Concretamente una rama denominada Deep Learning (DL), traducido como aprendizaje profundo. Dentro del DL vamos a destacar las redes neuronales artificiales (ANN) que van a ser las que usaremos.

Las ANN intentan aprender una función que se ajuste a la clasificación de un conjunto de datos. Para esto se realiza un proceso de entrenamiento que ajusta los parámetros de la función para reducir el error de la clasificación para dicho conjunto de datos. Esa es una de las razones por las que necesitamos muchos datos, por lo que debemos guardar la máxima cantidad de información posible y almacenarla de una manera que sea sencilla consultarla. Como nos encontramos en un campo complejo, ya que la información se encuentra en distintas fuentes, vamos a tener que hacer un proceso de combinación antes de almacenarla.

El trabajo se divide en dos partes, la primera es la recopilación de la máxima información posible sobre partidos jugados en la liga española, sobre sus jugadores, así como algunos factores que pueden influir en el resultado de un partido. La segunda parte consiste en aplicar algunos modelos estadísticos y modelos de machine learning para analizar si consiguen predecir los resultados.

Personalmente, yo soy no muy fan con lo que respecta al fútbol, pero la idea de construir un modelo predictivo a un problema de la vida real me genera una curiosidad por intentarlo.

1.1. Objetivos

El objetivo principal es el diseño e implementación de una Base de Datos (BD) sobre la liga española de fútbol. Para esto se pretende recopilar información de diversas fuentes, aplicando un proceso de unificación para poder almacenar todos los datos de una manera correcta. Se ha decidido usar un método de persistencia como es una BD porque de esta manera se pueden almacenar todos los datos sin duplicidades y organizados, y además extraer los datos que se deseen mediante consultas. Se ha decidido usar Python para la implementación porque facilita el trabajo al tener librerías ya implementadas.

Además se pretende aplicar un modelo estadístico para sacar la probabilidad de qué ocurra en un partido ,se intentará predecir el resultado de un encuentro entre dos equipos mediante algoritmos predictivos de machine learning y usando soluciones basadas en deep learning. Finalmente, se hará una comparativa entre los modelos que se implementen. Se ha decidido usar algoritmos de DL porque era una rama de dónde ya tenía conocimientos previos y para implementar un sistema inteligente.

Se podrían intentar predecir muchos otros campos, pero hemos decidido poner el punto final en estos dos sistemas, aunque tiene una escalabilidad extremadamente grande.

1.2. Estructura del trabajo

Para facilitar su lectura, el contenido de este documento se encuentra organizado en capítulos y secciones. A continuación se describe la estructura que vamos a seguir.

- ***Capítulo 1: Introducción*** ⇒ Capítulo introductorio que abarca los fundamentos del proyecto y describe los objetivos que se pretenden alcanzar.
- ***Capítulo 2: Estado del arte*** ⇒ Dedicada a proporcionar una base teórica general del ámbito en el que se mueve el problema a resolver, así como los objetivos a abordar.
- ***Capítulo 3: Metodologías y herramientas*** ⇒ Dedicada a proporcionar una descripción de las metodologías y herramientas utilizadas.
- ***Capítulo 4: Recopilación de datos*** ⇒ Puntos dedicados a describir el diseño e implementación de la BD y cómo hemos recopilado la información.
- ***Capítulo 5: Análisis de datos*** ⇒ Capítulo dedicado a analizar todos los datos almacenados en la BD anteriormente.
- ***Capítulo 6: Predicción de resultados*** ⇒ Capitulo dedicado a intentar aplicar algoritmos matemáticos para resolver problemas de predicción.
- ***Capítulo 7: Conclusiones*** ⇒ Detalla el conjunto de reflexiones obtenidas en cuanto al estudio de los resultados de las pruebas del proyecto, así como de posibles trabajos futuros a abordar.
- ***Capítulo 8: Anexo*** ⇒ Aborda APIs utilizadas, cómo he creado la BD, cómo ejecutar los scripts y tecnologías usadas.

2 Estado del arte

Las bases de datos nos han servido como sistema de persistencia, para almacenar información relacionada entre sí de manera correcta. Actualmente, existen varias bases de datos que contengan información sobre partidos de fútbol. Lo que no existen es una base de datos como la que vamos a crear que unifique información de distintas fuentes. Vamos a analizar las principales BD con datos sobre fútbol. Hemos escogido las BD más relevantes.

En la página [3] se pueden consultar los partidos de casi todas las ligas que se juegan en el mundo. De cada partido te da la información de los jugadores, los goles marcados, el resultado antes de medio tiempo y en el partido completo como mucha más información relevante en el partido, en la figura 2.1 se puede apreciar mejor.

En esta página [4] para cada equipo se pueden ver sus últimos partidos, tanto los que pertenecen a la competición de La Liga Santander (Liga entre los equipos de 1 división de España), como los jugados en otras competiciones. Da información del resultado en el descanso y el resultado al final del partido. En mi opinión te aporta muy poca información, ya que sólo tienes información de los goles anotados por cada uno de los equipos y en medio tiempo y final del partido. Por otro lado, te aporta información sobre partidos de otras competiciones que ocurren mientras se está disputando la Liga. Estos datos son importantes ya que influyen en el siguiente partido que vayan a jugar en la Liga, influyen en aspectos como el cansancio o si se están disputando algún partido importante de la otra competición.

API-Football [5] es una API de fútbol que nos proporciona información sobre los partidos de distintas ligas. Está pensada para apuestas ya que tiene un apartado que provee las cuotas antes del partido. En cuanto a información sobre el partido nos proporciona una lista de datos:

- Asistencias
- Disparos bloqueados
- Corners

Figura 2.1: Ejemplo de la página soccerbase.

League	Date	Home	Away	Res	H.T.	Comm.
PrimeraDiv	23.04.2019	Alaves	**Barcelona**	0-2	(0-0)	Full Time
PrimeraDiv	20.04.2019	**Barcelona**	Sociedad	2-1	(1-0)	Full Time
EuCL 1/4	16.04.2019	**Barcelona**	Man. Utd.	3-0	(2-0)	Full Time
PrimeraDiv	13.04.2019	Huesca	**Barcelona**	0-0	(0-0)	Full Time
EuCL 1/4	10.04.2019	Man. Utd.	**Barcelona**	0-1	(0-1)	Full Time
PrimeraDiv	06.04.2019	**Barcelona**	Atl. Madrid	2-0	(0-0)	Full Time

Figura 2.2: Ejemplo de la página footballresults.

- Contraataques
- Faltas
- Disparos de falta
- Goles

- Intento de goles
- Fuera de juego
- Posesión
- Tarjetas rojas
- Paradas del portero
- Disparos a puerta
- Tarjetas amarillas

Las apuestas de fútbol se han vuelto un mercado millonario, de acuerdo al paper [6], esta subida se debe a que actualmente tienen un sistema inteligente para predecir los resultados. En el pasado, las casas de apuestas usaban gente cualificada y modelos estadísticos muy complejos. Actualmente, la mayoría de las casas de apuestas están investigando en sistemas inteligentes de predicción de resultados para sacar beneficios y evitar riesgos financieros mejorando el porcentaje de acierto de predicción. En 2014, la compañía *Bing* de Microsoft predijo correctamente el resultado final de la ronda eliminatoria del mundial de fútbol y se convirtió en la primera casa de apuestas [7].

Los modelos bayesianos han dominado el aprendizaje supervisado en el negocio de apostar. Por ejemplo, el principio de aprendizaje Bayesiano fue usado para predecir los resultados de los partidos en Inlgaterra [8].

Vamos a analizar las principales aplicaciones actuales y estudios recientes sobre la predicción de resultados.

BeSoccer [9] es una aplicación para dispositivo móvil que contiene noticias de fútbol. Para los partidos te proporciona probabilidades de que gane uno, otro o haya un empate y estadísticas de los equipos. Además, existe un foro para discutir con otras personas sobre temas relacionados con todo lo anterior.

La ventaja que aprecio en este sistema, es que te sirve de apoyo para decidir. Te aporta información extra. Como la mayoría de sistemas basados en aprendizaje automático que no consiguen un porcentaje de acierto muy alto, sirven para dar apoyo a las decisiones de un humano.

Es un artículo de un blog [10] que intenta aplicar matemáticas para predecir resultados de la liga Inlgesa de la temporada 2018-19. Aplica la distribución de Poisson [11], que mide la probabilidad de que ocurra un número de eventos en un intervalo de tiempo si

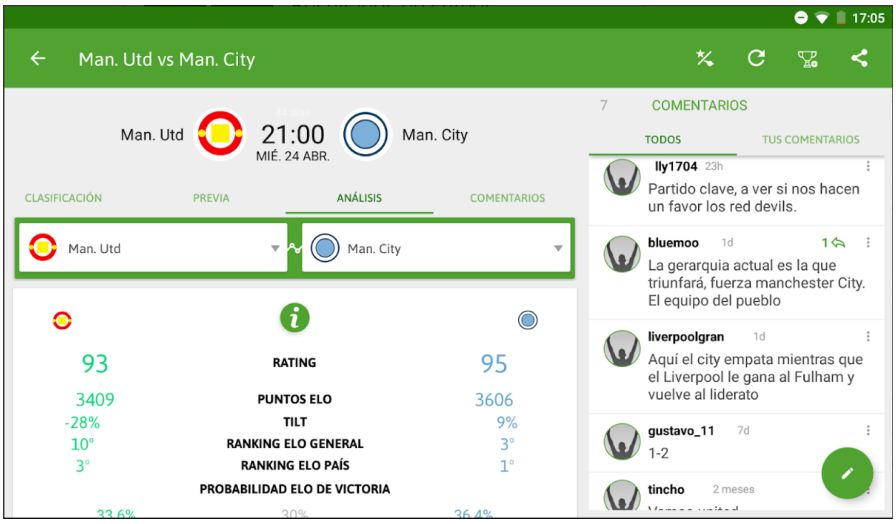

Figura 2.3: Ejemplo de la app BeSoccer.

esos eventos han ocurrido en un ritmo constante independientemente del tiempo ocurrido del último evento [10]. Considera un evento cuando se anota un gol, por lo tanto, en el tiempo de 90 minutos que dura un partido cada evento puede ocurrir un número de veces indeterminadas.

Como comenta, al final del artículo, como sólo se basa en los goles, el principal fallo de este sistema es de predecir un empate, ya que es muy difícil que coincidan dos equipos con el mismo número de goles. Aparte, para equipos que acaben de ascender no se tiene información sobre los goles, por lo que el sistema va a determinar que van a perder casi siempre.

Vamos a ver si su predicción final (Ver figura 2.4) de la temporada se acerca a cómo han quedado. A falta de 3 partidos para finalizar la temporada, se puede apreciar que los resultados se acercan, los que tienen que estar en las primeras posiciones lo están, los que tienen que estar por abajo en la clasificación también lo están. Hay algunos, como el equipo Wolves, que se le pronosticaba la 11ª posición y está en la 7ª. No es suficiente, se acerca pero no llega a acertar la posición exacta, te podría dar una estimación de la posición final en la clasificación pero no predecir el puesto exacto.

	Team	Points
1	Liverpool	89.94
2	Man City	84.66
3	Chelsea	77.82
4	Tottenham	77.39
5	Man United	75.48
6	Arsenal	73.97
7	Everton	52.71
8	Leicester	52.10
9	West Ham	49.39
10	Watford	48.08
11	Wolves	47.93
12	Bournemouth	47.58
13	Brighton	44.22
14	Southampton	44.02
15	Crystal Palace	40.64
16	Newcastle	38.65
17	Burnley	35.79
18	Fulham	31.76
19	Cardiff	28.83
20	Huddersfield	21.25

	EQUIPOS	PT
1	M. City	89
2	Liverpool	88
3	Tottenham H.	70
4	Chelsea	67
5	Arsenal	66
6	M. United	64
7	Wolverhampton Wanderers	51
8	Watford	50
9	Everton	49
10	Leicester City	48
11	West Ham Utd.	43
12	Crystal Palace	42
13	Newcastle United	41
14	Bournemouth	41
15	Burnley	40
16	Southampton	37
17	Brighton and Hove Albion	34
18	Cardiff City	31
19	Fulham	23
20	Huddersfield Town	14

Figura 2.4: Comparativa entre las predicciones y los resultados reales a falta de 3 jornadas para finalizar la competición.

3 Metodologías y herramientas

En esta sección voy a enunciar las metodologías y herramientas que he usado para llevar a cabo el trabajo. Voy a describir para qué me han servido y para qué las he utilizado.

3.1. Trello

He usado una metodología ágil scrum para organizarme y llevar el tiempo que he tardado en realizar cada tarea. He usado Trello que es un software de administración de proyectos con interfaz web y con cliente para iOS y android para organizar proyectos.

`https://trello.com/`

Figura 3.1: Logo Trello.

A la hora de usar una metodología ágil, al ser una persona sólo no he necesitado organizar el tablero muy exhaustivamente.

He creado 4 listas, en la primera sólo existen 3 tarjetas y en ellas apunto el tiempo que tardo para cada apartado del proyecto, esta parte es más para mí, para saber si hay alguna parte que me cueste más o me atasque y así poder mejorar. La siguiente lista enumera las tarjetas que quedan por hacer, como su propio nombre indica, en ella pongo las tareas que aún no he realizado. La penúltima tarjeta es para las tareas que estoy realizando y aún no están terminadas, y la última agrupa todas las tarjetas que ya han sido realizadas por si hubiera algún problema y tuviera que devolverla a la fase anterior.

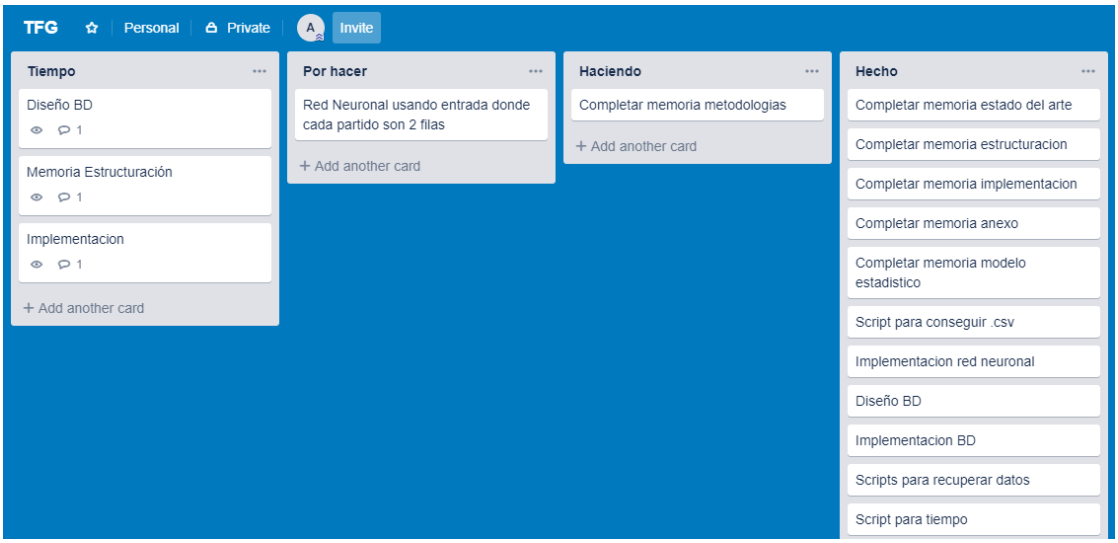

Figura 3.2: Tabla Trello.

3.2. Draw.io

Para diseñar el diagrama UML he utilizado una aplicación llamada Draw.io. Es un software que te permite dibujar diagramas de UML,ER, diagramas de proceso, etc.

`https://www.draw.io/`

Figura 3.3: Logo Draw.io.

3.3. MySQL

MySQL es un sistema de gestión de bases de datos relacional. Lo hemos usado para implementar la estructura de la Base de Datos y su gestión.

Url: `https://www.mysql.com/`

Figura 3.4: Logo MySQL.

3.4. Python

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional.

`https://www.python.org/`

En computación, la palabra script se usa para referirse a un archivo que contiene una secuencia lógica de órdenes o un archivo de procesamiento por lotes. Este suele ser un programa simple, almacenado en un archivo de texto plano.

Los scripts siempre son procesados por algún tipo de intérprete, que es responsable de ejecutar cada comando de forma secuencial. Python contiene un intérprete que nos permite ejecutar scripts. Hemos implementado scripts en python y hemos usado el intérprete para crear la BD y rellenarla.

Figura 3.5: Logo Python.

3.5. Latex

`https://www.latex-project.org/`

Figura 3.6: Logo Latex.

LaTeX es un sistema de composición tipográfica de alta calidad. Incluye características diseñadas para la producción de documentación técnica y científica. LaTeX es el estándar para la comunicación y publicación de documentos científicos. Existen plantillas de trabajos para hacer más rápida la creación de documentos comunes como trabajos de fin de grado, currículums, etc.

De Latex usamos Overleaf que es un editor de textos online de Latex que nos facilita la organización y calidad del documento.

3.6. GitHub

`https://github.com/`

GitHub es un sistema en la nube que permite a los desarrolladores llevar un control de versiones, es decir, permite almacenar código y llevar un control y registro de de cualquier cambio sobre este código.

También lo hemos usado como sistema de seguridad por si ocurriera algún fallo en mi ordenador que no me dejara recuperar los ficheros.

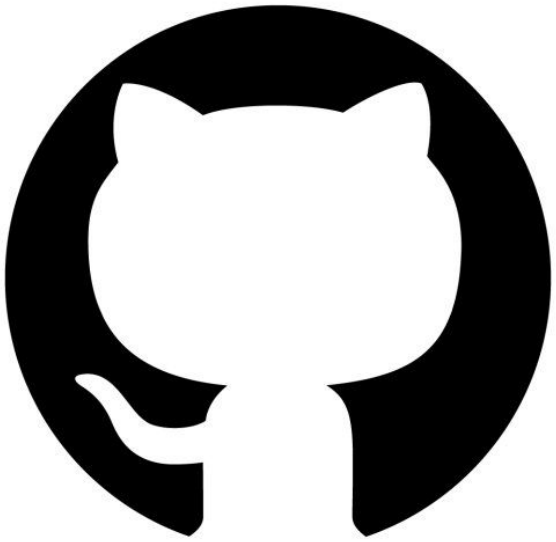

Figura 3.7: Logo GitHub.

4 Recopilación de datos

4.1. Introducción

Para resolver el problema de predicción de resultados deportivos no había ninguna Base de Datos (BD) con todos los datos que queríamos, por lo que hemos tenido que diseñar y rellenar una BD en MySQL (Sistema de gestión de bases de datos) para unificar la información de diferentes procedencias y, además, para poder mantener la integridad de los datos y su persistencia.

Los principales campos en los que nos hemos centrado son: los jugadores, los equipos y los partidos. De los jugadores queremos guardar sus atributos físicos en el campo, como la velocidad, la resistencia, la calidad del disparo a puerta, etc. De cada equipo queremos guardar los puntos que lleva en la jornada que se va a jugar el partido así como la posición en la clasificación. Del partido queremos guardar todos los datos que sean posibles, en los siguientes capítulos ya analizaremos la información, pero actualmente queremos recopilar la máxima información posible. Por lo tanto, del partido se han almacenado información básica de los goles y el resultado, y además hemos incluido información externa al partido pero que influye en él, como es la temperatura a la que se está disputando el encuentro entre los dos equipos.

En la figura 4.1 se muestra el diagrama de clases UML de la base de datos diseñada. Por claridad no se muestran todos los atributos de las tablas, pero en la siguiente sección analizaremos los más importantes.

Para realizar el diseño del diagrama UML hemos tenido en cuenta un error típico en bases de datos de programas de facturación, ya que hemos necesitado añadir los atributos que pueden variar en el tiempo como atributos de la relación. Es decir, hemos añadido dos tablas, una para equipo y otra para los jugadores, para almacenar los datos que pueden cambiar. Por ejemplo, un jugador puede jugar en un equipo pero a la temporada siguiente que lo fiche otro distinto, y en este caso tendría que cambiar su campo equipo, por lo que la información de cuando jugaba en el equipo anterior se perdería. Para ello hemos creado las tablas intermedias *EquipoPartido* y *JugadorPartido* como atributos de la relación para así poder almacenar estos cambios.

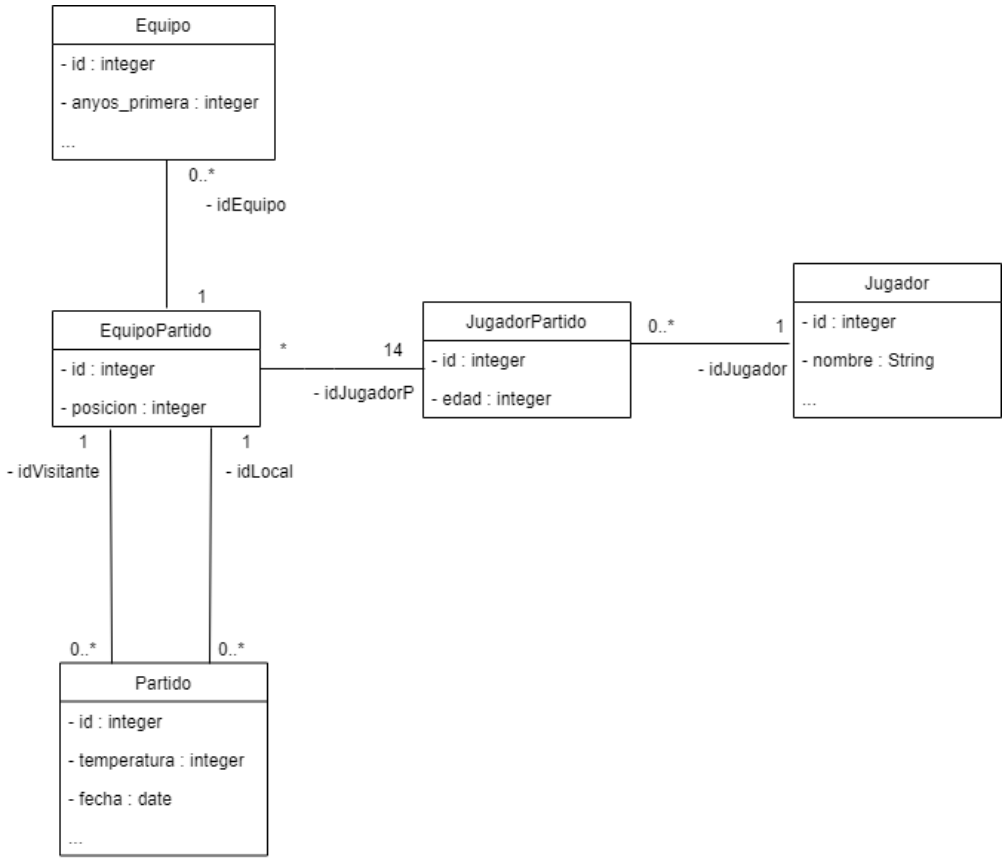

Figura 4.1: Diagrama UML de la BD.

Para la recopilación de toda estos datos se han utilizado diversas fuentes. En la tabla 4.1 se muestra un listado de las mismas.

Fuente	**Descripción**
`https://espanol.wunderground.com/history/`	Temperatura máx y mín
`https://www.kaggle.com/ricardomoya/football-matches-of-spanish-league`	Partidos desde 1970
`http://www.football-data.co.uk/spainm.php`	Partidos desde 2004
`https://www.kaggle.com/karangadiya/fifa19`	Datos sobre jugadores 2019
`https://www.kaggle.com/abhilash04/fifa-18-player-ratings`	Datos sobre jugadores 2018

Tabla 4.1: Listado de las diversas fuentes.

4.2. Tabla Partido

Por simplificar el diagrama anterior, sólo se han incluido las claves primarias y algún atributo de la tabla. El listado completo de atributos de la entidad *Partido* se puede consultar en la tabla 4.2.

Esta es la tabla principal, sus campos más importantes son: temporada, fecha y hora, los cuales permiten distinguir un partido de otro jugado por los mismos equipos. También se incluyen los goles marcados por ambos equipos y el resultado que se obtiene de los goles (explicado en la introducción). Para almacenar toda la información se ha implementado un script en Python que lee de un CSV todos estos campos y los introduce en la tabla Partido.

Para obtener la información se han empleado dos fuentes, la primera contiene partidos desde 1970, pero sólo almacena los goles marcados y el *timestamp* (muy útil e importante), a parte de los campos básicos como los nombres de los equipos y la temporada. La otra fuente utilizada contiene partidos desde 2004, pero en este caso sí que incluye el resto de campos mencionados en la tabla 4.2.

Las fuentes utilizadas para recopilar estos datos han sido:

- `https://www.kaggle.com/ricardomoya/football-matches-of-spanish-league`
- `http://www.football-data.co.uk/spainm.php`

4.3. Tabla Jugador

En la tabla 4.3 se almacenan los atributos de la entidad *Jugador*. Todos los atributos están puntuados de 0 a 100, siendo 0 el peor valor posible y 100 el mejor. Las calificaciones de los jugadores se han obtenido de una base de datos donde FIFA los ha calificado. Esta BD sólo contiene las calificaciones de los años 2018 y 2019.

Por motivos de inexistencia de información, sólo se han guardado la información de los jugadores de 2018 y 2019, los campos que vamos a destacar son la media, el tipo de cuerpo y la velocidad del jugador. Para guardar estos datos se han sacado de Kaggle. En el dataset de 2018 hay menos campos que en el de 2019, por lo que algunos atributos pueden aparecer vacíos.

Las fuentes utilizadas para recopilar estos datos han sido:

Nombre	Tipo de dato	Descripción
Temporada	String	Temporada en la que se ha jugado
Fecha	Date	Fecha en formato dd/mm/yyyy
Hora	Integer	Hora de comienzo del partido
Temperatura	Integer	Valor de la temperatura en grados
Local	String	Nombre del equipo local
Visitante	String	Nombre del equipo visitante
FTHG	Integer	Número de goles en todo el partido del equipo local
FTAG	Integer	Número de goles en todo el partido del equipo visitante
FTR	String	Resultado del partido (H=Local, D=Empate, A=Visitante)
HTHG	Integer	Número de goles del equipo local en mitad de tiempo
HTAG	Integer	Número de goles del equipo visitante en mitad de tiempo
HTR	String	Result en el descanso (H=Local, D=Empate, A=Visitante)
Referee	String	Nombre del árbitro
HS	Integer	Disparos equipo local
AS	Integer	Disparos equipo visitante
HST	Integer	Disparos equipo local a puerta
AST	Integer	Disparos equipo visitante a puerta
HC	Integer	Córners equipo local
AC	Integer	Córners equipo visitante
HF	Integer	Faltas equipo local
AF	Integer	Faltas equipo visitante
HY	Integer	Tarjetas amarillas equipo local
AY	Integer	Tarjetas amarillas equipo visitante
HR	Integer	Tarjetas rojas equipo local
AR	Integer	Tarjetas rojas equipo visitante

Tabla 4.2: Atributos de Partido.

- `https://es.fifa.com/`
- `https://www.kaggle.com/karangadiya/fifa19`
- `https://www.kaggle.com/abhilash04/fifa-18-player-ratings`

Otro problema encontrado es que, al usar diferentes fuentes para obtener datos de los equipos, sus nombres se pueden encontrar de distintas maneras. Por ejemplo, podemos encontrar *FC Barcelona* o *Barcelona* según la fuente de datos utilizada. La solución

Nombre	Tipo de dato	Descripción
Nombre	String	Nombre del jugador
País	String	Nacionalidad del jugador
Anyos	Integer	Años del jugador
Media	Integer	Media del jugador
Potencial	Integer	Fuerza
Equipo	String	Equipo en el que juega
Aceleracion	Integer	Valor de la aceleración
Agresividad	Integer	Valor de la agresividad
Agilidad	Integer	Valor de la agilidad
Balance	Integer	Valor del balance
Control_balon	Integer	Control del balón
BodyType	String	Complexión física
Compostura	Integer	Compostura
Cruce	Integer	Cruce
Giro	Integer	Giro
Regate	Integer	Regate
Acabado	Integer	Finalización
Falta_acc	Integer	Precisión tiro de faltas
Fuerza	Integer	Fuerza
Visión	Integer	Visión

Tabla 4.3: Atributos de Jugador.

ha sido añadir un campo en la tabla equipo que se llama "Alias", el cual contiene los distintos nombres utilizados para el mismo equipo. Esto hay que tenerlo en cuenta a la hora de hacer una consulta a la tabla, ya que hay que comparar con el alias por si acaso el nombre no fuera el más común del equipo (que es el que está puesto como nombre principal).

4.4. Temperatura

También hemos querido obtener la temperatura a la que se jugó el partido. No hay ninguna API gratuita que te permita saber la temperatura de un día a una hora exacta, sólo existen datos históricos de temperaturas máximas y mínimas por día.

URL de donde hemos obtenido los datos: `https://espanol.wunderground.com/history/`

Para obtener la temperatura máxima y mínima de un día, hemos implementado un script que realiza consultas a una página web, va cambiando la fecha y el lugar, realiza

la consulta y se guarda la temperatura.

Dado que solo disponíamos de la temperatura máxima y mínima, hemos creado una función de distribución para obtener la temperatura a la hora que se jugó el partido. Para esto hemos usado la función coseno para ajustar los valores (ver figura 4.2). Hemos usado este método ya que buscábamos la temperatura a una hora y al no encontrarla, hemos querido interpolar de una forma sencilla y obtener la función que nos da la temperatura a una hora exacta.

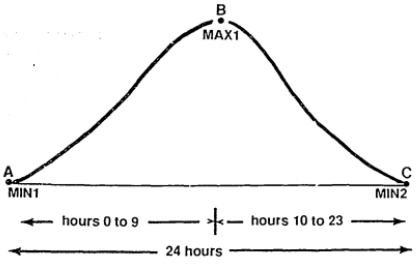

Figura 4.2: Ilustración de la función coseno que ajusta el máximo y el mínimo

La respuesta obtenida utiliza un formato de hora distinto al habitual, las 00:00 se representan por 0000 y las 14:00 serían las 1400, ya que al obtener la hora del timestamp es más sencillo guardarlo de esta manera.[12]

Para interpolar se ha usado el método de ajuste del coseno, la temperatura mínima suele ocurrir al 05:00 del tiempo estándar local y la máxima a las 14:00. Por conveniencia en los cálculos, el comienzo lo hemos puesto a 0. Por lo tanto, las horas entre 05:00 y 14:00, ahora son las horas entre 00:00 y 9:00 que es el segmento AB (ver figura 4.2), y el resto de horas están en el segmento BC.

Las series de Fourier constituyen la herramienta matemática básica del análisis de Fourier empleado para analizar funciones periódicas a través de la descomposición de dicha función en una suma infinita de funciones sinusoidales mucho más simples (como combinación de senos y cosenos con frecuencias enteras) [13].

El método de ajuste de coseno ajusta a una serie de Fourier cada mitad de la diagonal de la curva, es decir, desde MIN1 hasta MAX1 sería una diagonal, y la otra desde MAX1 hasta MIN2. Para entenderlo mejor lo hemos descrito como una transformación lineal. Una variable (x) se puede definir en términos de la hora (hr), así es como la temperatura se convierte en una función lineal donde:

$$x = cos(hr * \frac{\pi}{9})$$

para las horas (hr) de 0 a 9, y

$$x = cos[(hr - 10) * \frac{\pi}{13}]$$

para las horas de 10 a 23.

Usando la ecuación para la recta $y = mx + b$ y la gráfica 4.2, la pendiente para el segmento AB sería:

$$-(\frac{MAX1 - MIN1}{2})$$

y para el segmento BC:

$$\frac{MAX1 - MIN2}{2}$$

Se ha asumido que el valor b para ambos segmentos es dado por la mitad del valor de la amplitud. Combinando los términos anteriores, finalmente queda la siguiente ecuación para calcular la temperatura:

$$T(hr) = -(\frac{MAX1 - MIN1}{2}) * cos(hr * \frac{\pi}{9}) + (\frac{MAX1 + MIN1}{2})$$

para las horas de 0 a 9, y

$$T(hr) = -(\frac{MAX1 - MIN2}{2}) * cos[(hr - 10) * \frac{\pi}{13}] + (\frac{MAX1 + MIN2}{2})$$

donde $T(hr)$ es la temperatura para las horas entre 10 y 23. Después de los cálculos hay que volver a transformar la hora al tiempo estándar, añadiendo 5 horas.

Por ejemplo, supongamos que se ha jugado un encuentro entre el equipo A (local) contra el equipo B (visitante) el día 06/10/1997 a las 18:00 (UTC). Usamos el script para obtener la temperatura mínimo y máxima del día 06/10/1997, y nos indica que la temperatura máxima para ese día fue de 27ºC y la temperatura mínima de 16ºC. Cómo la hora a la que se ha jugado el partido fue a las 18:00, que está entre las 10:00 y las 23:00, sustituimos en la segunda ecuación:

$$T(18) = -(\frac{27-16}{2}) * cos[(18-10) * \frac{\pi}{13}] + (\frac{27+16}{2})$$

Resolvemos la ecuación y nos daría T(18)=21.5003

Finalmente, aplicamos un redondeo y nos quedaría que la temperatura sería de 22ºC.

4.5. Estadísticas

Después de haber recopilado toda la información posible, en esta sección vamos a mostrar algunos datos estadísticos de la BD. En la tabla 4.4 se puede ver un resumen del número de registros almacenados para cada tabla así como el tamaño que ocupan.

Nombre de tabla	**Filas**	**Tamaño**
Equipo	53	16KB
EquipoPartido	34.758	10MB
Jugador	520	128KB
JugadorPartido	492	80 KB
Partido	11.950	2MB

Tabla 4.4: Estadísticas de la BD.

5 Análisis de datos

Una vez finalizada y rellenada la BD, vamos a obtener algunas estadísticas sobre los partidos jugados en los últimos años y de como influye la posición en la clasificación a la hora de jugar un partido.

En primer lugar, vamos a mostrar diferentes estadísticas sobre la recopilación de datos llevada a cabo. En total hemos recopilado 11,950 partidos de 53 equipos distintos. Para obtener los distintos gráficos vamos a usar scripts de Python para hacer querys a la BD. Por tener una muestra representativa, vamos a usar los partidos desde la temporada 2004-05 hasta la temporada 2017-18, ambas incluidas. Por lo que el número de muestras se nos quedará en 4940 partidos y 40 equipos.

Las estadísticas nos ayudarán a saber que datos tenemos mayoritariamente. Vamos a ver si se ganan más partidos jugando como equipo local o como visitante. De los 4940 partidos seleccionados, 1498 serían victorias del equipo visitante, 2553 serían victorias del equipo local y 1269 serían partidos empatados. En la figura 5.1 se puede ver una representación gráfica de estos datos.

Ahora vamos a desglosar estos mismos datos por temporadas para ver si hay alguna temporada en la que se haya producido alguna anomalía. En la figura 5.2 se puede ver un gráfico de barras que muestra el total de victorias jugando como local, visitante y empatados por temporada.

Como podemos observar en el gráfico, más o menos todas las temporadas siguen el mismo patrón, el equipo visitante gana más partidos que empates, además, entre los partidos ganados como visitante y los empatados, suman los ganados por el equipo local, es decir, estos representan aproximadamente el 50 % de los resultados.

De este gráfico podemos sacar algunos datos objetivos, como que el número de victorias jugando como local está en el rango [197,162], jugando como equipo visitante el rango es [118,88] y de empatados es [105,79].

En la figura 5.3 se muestra también los goles que se marcan por temporada, separándolos cuando se juega como visitante y cuando se juega como local. Algunas objeciones de estas estadísticas es que los goles en una temporada están en el rango [641,518] y cuando

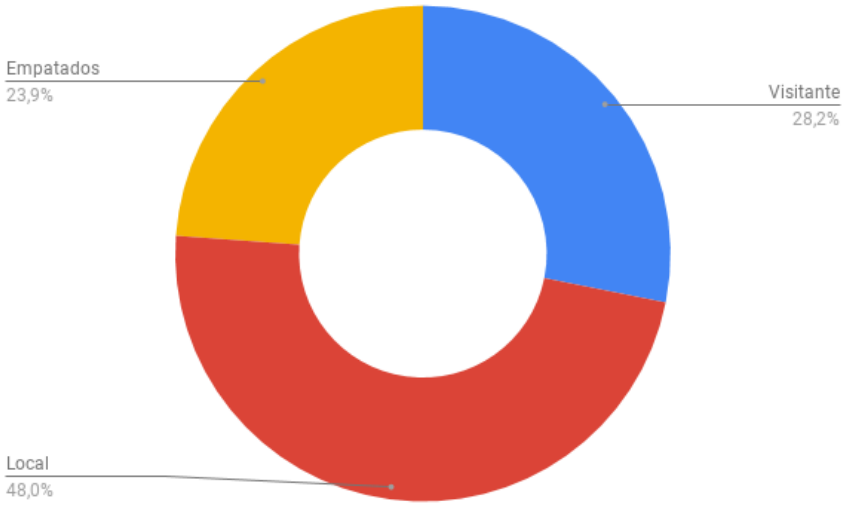

Figura 5.1: Gráfico que muestra el porcentaje de victorias jugando como local, visitante y empatados.

se juega como equipo visitante [486,404].

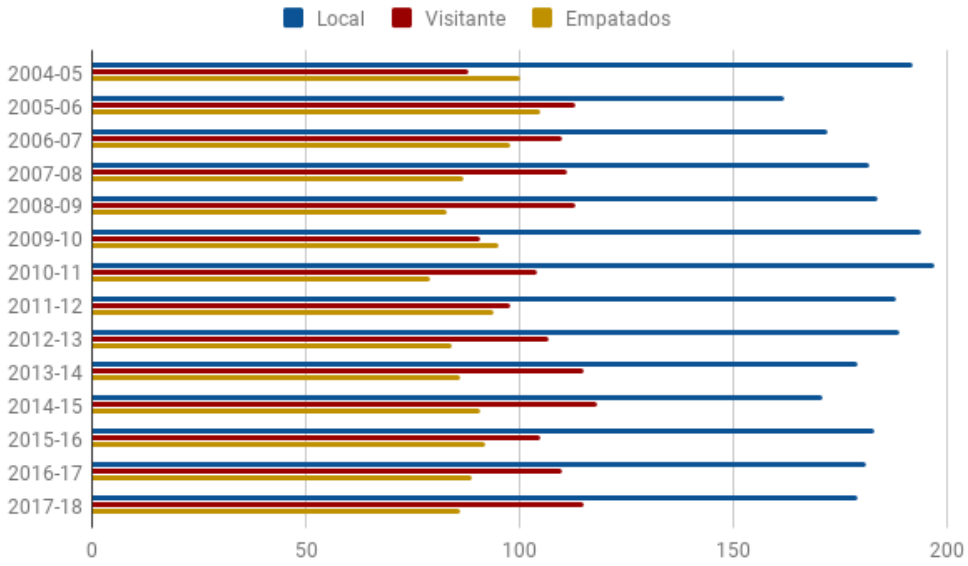

Figura 5.2: Gráfico de barras que muestra el total de victorias jugando como local, visitante y empatados por temporada.

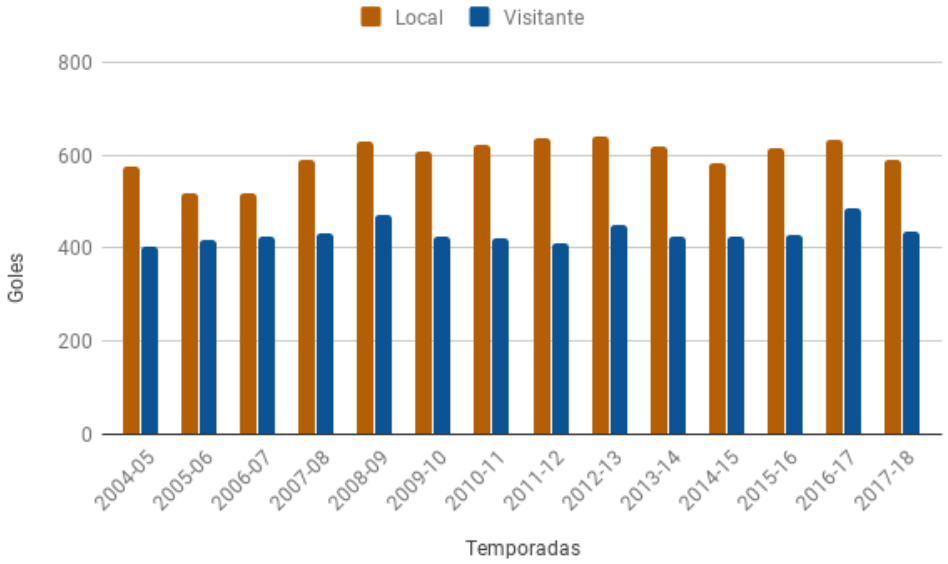

Figura 5.3: Número de goles por temporada, dividido cuando se juega como local y visitante.

6 Predicción de resultados

Después de haber analizar las estadísticas de la información almacenada en la BD en el capítulo anterior, vamos a poner un punto de partida que va a ser un modelo estadístico, a partir de él, se intentará mejorar utilizando algoritmos de aprendizaje automático supervisado para intentar predecir el resultado de un evento entre dos equipos.

6.1. Modelo Estadístico

El primer modelo que vamos a usar es un modelo estadístico, el cual permite ver, en el enfrentamiento de dos equipos, la probabilidad que gane uno u otro en función de los resultados anteriores.

Este modelo está implementado usando un script de Python que recibe los dos equipos que se van a enfrentar y devuelve una probabilidad de que gane uno, otro o de que empaten.

Vamos a dividir el modelo en dos para que diferencie cuando un equipo está jugando como local o como visitante, sólo tiene en cuenta si en eventos anteriores de los mismos equipos ha ganado, perdido o empatado. En la figura 6.1 se puede ver un ejemplo, en el que nos muestra los porcentajes que se obtiene de aplicar el modelo estadístico del encuentro del Real Madrid contra el Barcelona, teniendo en cuenta que sólo se usan los partidos desde la temporada 2004 hasta la del 2017 y cuando el Barcelona está jugando como local.

Por otro lado, cuando en el encuentro Barcelona vs Real Madrid, el Barcelona está jugando como visitante (en el mismo conjunto de datos que antes, 2004 hasta 2017), se muestra en la figura 6.2.

El modelo estadístico devuelve los porcentajes de victoria, derrota o empate de cada equipo. Su funcionamiento consiste en contar todas las veces que ha ganado, perdido o empatado cada equipo y dividirlo por las veces que se han enfrentado.

Una vez realizado el modelo, hemos usado para entrenar los partidos desde la tempo-

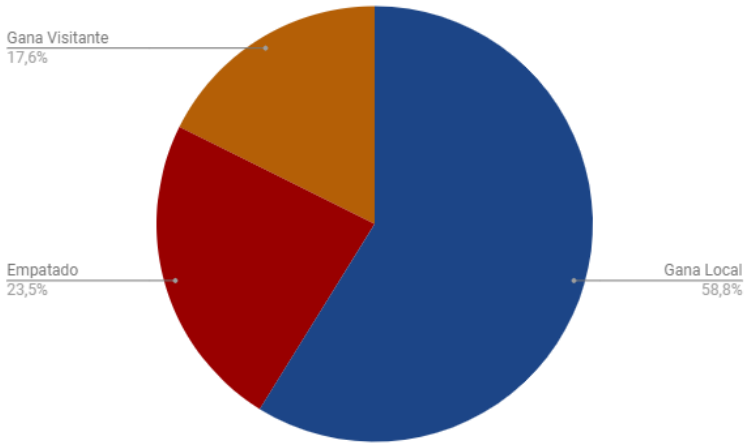

Figura 6.1: Ejemplo de solución del modelo estadístico cuando el Barcelona juega como local contra el Real Madrid.

rada 2004-05 hasta la de 2016-17 para entrenar(train), y dejando la de 2017-18 para ver cuantos acierta(test). De cada partido obteníamos 3 posibilidades, que gane el equipo local, que empaten o que gane el equipo visitante. Nos quedamos con la que sea mayor de las tres y se la aplicamos a los nuevos partidos de la parte de test. Haciendo caso nada más de esta estadística, de 380 partidos acertaríamos el resultado de 161, que es el 42.36 % de acierto.

6.2. Modelo ML

Después de haber obtenido un resultado usando un modelo estadístico, vamos a ver si usando modelos de ML podemos mejorar el porcentaje de acierto que hemos obtenido.

El principal objetivo del aprendizaje supervisado es generalizar sobre un conjunto de datos etiquetado para poder predecir nuevas muestras. Es decir, el algoritmo ajusta un modelo (aprende) a partir de un conjunto de datos de entrenamiento del que se dispone de la solución (la etiqueta a predecir), para esto modifica los parámetros del modelo intentando reducir el error del resultado de la predicción. Posteriormente este modelo se aplica sobre un conjunto de datos de evaluación que no se han visto durante el entrenamiento. Vamos a usar redes neuronales para resolver este problema (Consultar anexo sobre redes neuronales).

Lo primero que hemos realizado ha sido obtener todos los encuentros entre la temporada 2004-05 y 2017-18 con una query sobre la base de datos. Además hemos utilizado las temporadas 2004-05 a la 2016-17 para el entrenamiento del modelo, y la temporada

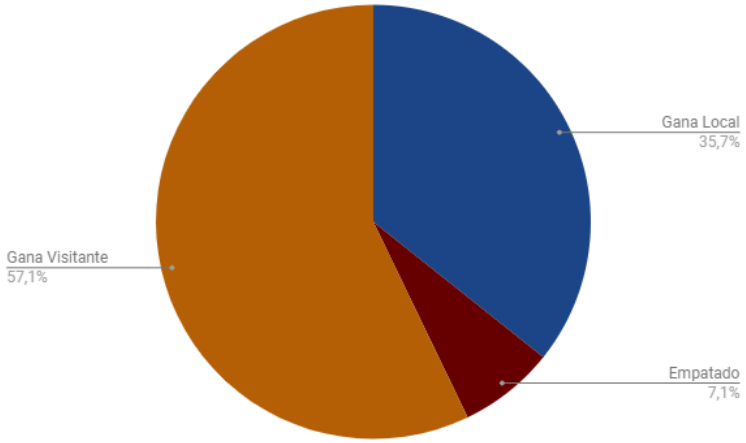

Figura 6.2: Ejemplo de solución del modelo estadístico cuando el Barcelona juega como visitante contra el Real Madrid.

2017-18 para la fase de test, para comprobar que el modelo está generalizando correctamente y no está ocurriendo overfitting (se adapta sólo a los datos de entrenamiento y no generaliza).

6.3. Dataset

Como estamos usando aprendizaje supervisado, para que el modelo aprenda necesitamos pasarle unos datos como entrada que van a ser los que use para predecir el resultado. De la BD que tenemos se pueden sacar multitud de datasets y combinaciones de atributos posibles. Pero vamos a obtener uno que muestre las principales características que influyen en un partido. Para esto usaremos los puntos que llevaba en las 5 jornadas anteriores y el número de jornada en la que se encuentra, por lo que vamos a obtener un dataset uniendo la información comentada.

Para conseguir el máximo número de muestras posibles, hemos realizado una "división"de partidos. Ahora hemos dividido esa información en dos distintas, una con el equipo visitante, los partidos anteriores y el número de jornada y otro con el equipo visitante. Hemos añadido otra columna llamada *lv* que indica con un 1 si el equipo esta jugando como local o como visitante. De este modo, vamos a tener dos filas para el mismo partido.

La división que hemos realizado nos lleva a tener un total de 10641 muestras. Vamos a usar los partidos de la temporada 2017-18 para el test, para comprobar que nuestra

algoritmo está aprendiendo correctamente.

- *equipo*: Nombre del equipo
- *lv*: Contiene un 1 si está jugando como local, 0 en otro caso
- *temporada*: Temporada en la que se están disputando los partidos
- *jornada5l,jornada4l,jornada3l,jornada2l,jornada1l*: Puntuación en las jornadas anteriores
- *numJornada*: Número de jornada que se está jugando
- *ganaLocal*: Contiene un 1 si ha ganado el equipo local, 0 en otro caso
- *ganaVisitante*: Contiene un 1 si ha ganado el equipo visitante, 0 en otro caso
- *empate*: Contiene un 1 si han empatado, 0 en otro caso

	equipo	lv	temporada	jornada5l	jornada4l	jornada3l	jornada2l	jornada1l	numJornada	ganaLocal	empate	ganaVisitante
0	Ath Madrid	1	2004-05	0	0	0	0	0	1	1	0	0
1	Malaga	0	2004-05	0	0	0	0	0	1	1	0	0
2	Espanol	1	2004-05	0	0	0	0	0	1	0	1	0
3	La Coruna	0	2004-05	0	0	0	0	0	1	0	1	0
4	Numancia	1	2004-05	0	0	0	0	0	1	0	1	0
5	Betis	0	2004-05	0	0	0	0	0	1	0	1	0

Figura 6.3: Estructura del dataset.

Al final del fichero se incluyen las 3 columnas que contienen los resultados (o etiquetas a predecir): ganaLocal, ganaVisitante y empate. En el aprendizaje supervisado es necesario tener las soluciones para que el modelo pueda entrenar. Como se ha explicado, esta tabla contiene un 1 en la fila de *ganaLocal* si ha ganado el encuentro el equipo local, un 1 en *ganaVisitante* si ha ganado el equipo visitante, y un 1 en *empate* si el encuentro ha acabado en empate.

Se ha empleado una red pequeña para intentar entrenarla ya que la entrada es sencilla y así nos aseguramos que no hará overfitting.

Para el entrenamiento del modelo se han usado 1000 épocas para intentar entrenar el algoritmo, pero la muestra tiene mucho ruido por lo que le cuesta aprender y no se consigue llegar a un buen resultado. 6.6

ganaLocal	ganaVisitante	empate
1	0	0
0	0	1
0	0	1
0	1	0

Figura 6.4: Soluciones en fichero .csv.

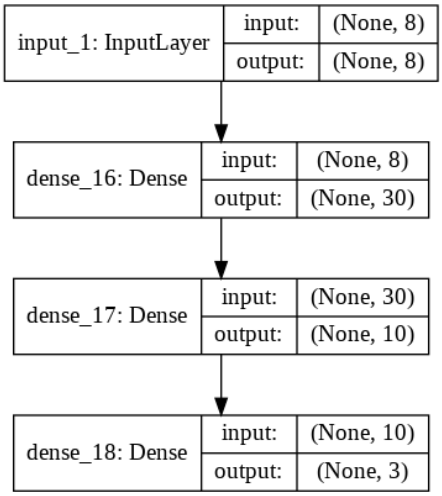

Figura 6.5: Estructura de a Red Neuronal usada para el dataset.

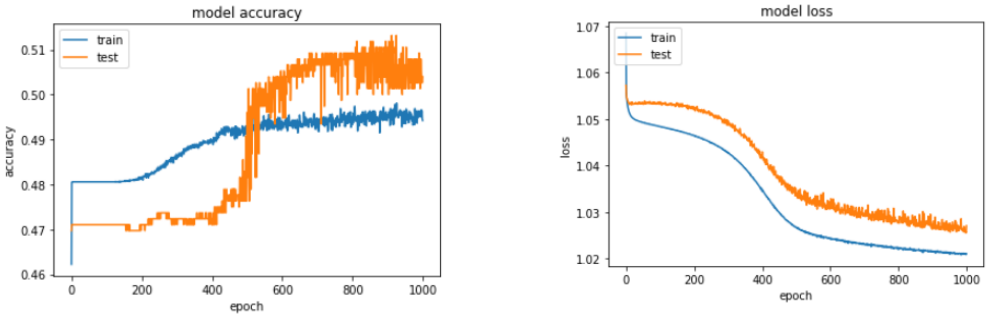

Figura 6.6: Gráfica de los valores de loss y accuracy durante el entrenamiento con el dataset.

6.4. Dataset2

En este caso, hemos unido la información de un partido en uno sólo en vez de dividirla. Porque de la otra manera perdíamos información al no saber contra quien se está jugando. El dataset estará formado por 14 campos que se definen a continuación.

- *equipoLocal*: Nombre del equipo que juega en casa
- *equipoVisitante*: Nombre del equipo que juega como visitante
- *temporada*: Temporada en la que se están disputando los partidos
- *jornada5l,jornada4l,jornada3l,jornada2l,jornada1l*: Puntuación en las jornadas anteriores
- *visitante5l,visitante4l,visitante3l,visitante2l,visitante1l*: Puntuación en las jornadas anteriores
- *numJornada*: Número de jornada que se está jugando
- *ganaLocal*: Contiene un 1 si ha ganado el equipo local, 0 en otro caso
- *ganaVisitante*: Contiene un 1 si ha ganado el equipo visitante, 0 en otro caso
- *empate*: Contiene un 1 si han empatado, 0 en otro caso

Esta misma información en un fichero .csv se podría apreciar así:

	equipoLocal	equipoVisitante	temporada	jornada5l	jornada4l	jornada3l	jornada2l	jornada1l	jornada5v	jornada4v	jornada3v	jornada2v	jornada1v	numJornada	ganaLocal	ganaVisitante	empate
0	Ath Madrid	Malaga	2004-05	0	0	0	0	0	0	0	0	0	0	1	1	0	0
1	Espanol	La Coruna	2004-05	0	0	0	0	0	0	0	0	0	0	1	0	0	1
2	Numancia	Betis	2004-05	0	0	0	0	0	0	0	0	0	0	1	0	0	1
3	Mallorca	Real Madrid	2004-05	0	0	0	0	0	0	0	0	0	0	1	0	1	0
4	Osasuna	Ath Bilbao	2004-05	0	0	0	0	0	0	0	0	0	0	1	0	0	1

Figura 6.7: Fichero .csv visto de forma gráfica.

Para introducir los equipos en la red neuronal hemos usado una capa de *Embedding* [14], típica capa que es usada por las redes de reconocimiento de sentimientos y que analizan texto. Requiere que los datos que se le pasen estén codificados, por eso cada equipo lo hemos representado como un número en un diccionario. Se ha creado una pequeña función que cambia el nombre del equipo por su id del diccionario.

El modelo que hemos usado, implementado con el API de Keras, es una red neuronal que contiene 3 inputs (ver figura 6.8). El primer input es el equipo local, pero como la

red neuronal no admite cadenas, para poder entrenar hemos convertido todos los equipos a un id, convirtiéndolos a un valor entero con un diccionario de clave-valor. El segundo input es el nombre del equipo visitante que sigue el mismo curso que el anterior. El último input es el dataset comentado anteriormente, con los 11 valores de los 5 últimos partidos jugados por el equipo y otro campo que es el número de jornada.

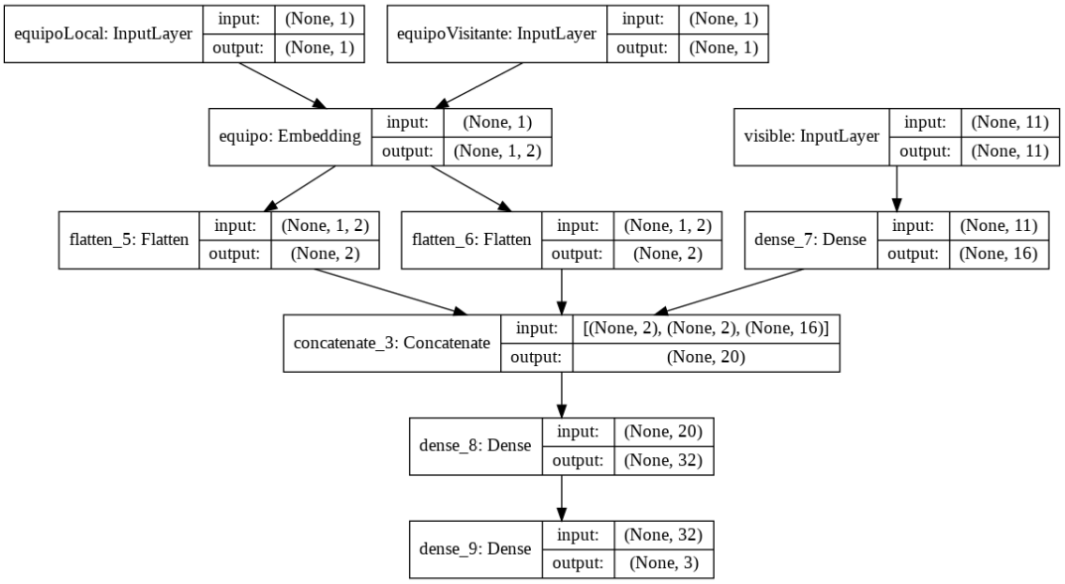

Figura 6.8: Estructura del modelo.

Hay una capa intermedia que une las tres entradas en una sola capa realizando una operación de concatenación. Esta capa simplemente coloca los inputs que recibe uno detrás de otro y crea una capa con la dimensión de la suma de los inputs. Por ejemplo, en nuestro caso sería la suma de la capa de Embedding por parte del equipo local (2), la capa de Embedding por parte del equipo visitante (2) y los 11 valores que hemos pensado que son importantes para determinar el resultado de un evento. Después de esta capa se añaden otras 3 capas planas (capas densas o fully conected) para computar la salida.

Este modelo se entrena durante 1000 épocas para cada conjunto de entrenamiento. Incluimos el conjunto de test en el entrenamiento para ver si esta ocurriendo overfitting. En la figura 6.9 se pueden ver los resultados durante el entrenamiento. El eje horizontal representa las épocas de entrenamiento, y el eje vertical representa la pérdida (loss) o error en la gráfica de la izquierda y el accuracy (acc) en la gráfica de la derecha. La línea azul muestra el resultado para el conjunto de entrenamiento y la línea naranja para el

conjunto de test.

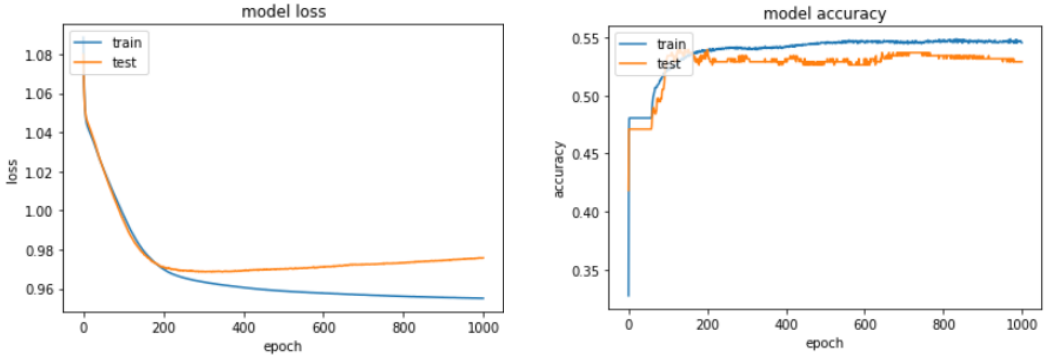

Figura 6.9: Gráfica de los valores de loss y accuracy durante el entrenamiento usando el segundo dataset.

Como se puede ver en la imagen, el conjunto de datos tiene que tener mucho ruido porque el modelo va adaptándose poco a poco al ruido y los resultados oscilan mucho, pero no consigue generalizar correctamente. Se consigue un poco más del 50 % que es un porcentaje muy bajo aunque sea sólo para guiar a un humano, sin llegar a tomar decisiones de forma autónoma.

Con la ayuda de la API de keras, la capa de Embedding que contiene a los equipos se puede representar en dos dimensiones y ver las diferencias o similitudes que existen entre los diferentes equipos, al no haber conseguido un buen aprendizaje los equipos se encuentran mezclados entre ellos, aunque se puede apreciar como el Real Madrid se encuentra cerca del Barcelona, equipos que en principio tienen comportamientos similares a lo largo de una temporada 6.10.

6.5. Normalización

Hemos normalizado los datos de entrada a la red (ver figura 6.11), ya que es normal escalarlos porque la red tiene mejor rendimiento cuando los valores de entrada están entre 0 y 1 (ver figura 6.12). Para ello, hemos usado los métodos que proporciona Keras, hemos importado la clase MinMaxScaler y los métodos fit_transform y tranform.

6.6. Comparativa

Después de haber obtenido los resultados de los 3 modelos que se han probado, vamos a hacer una comparativa entre ellos para ver cual es el que mayor acierto nos proporciona

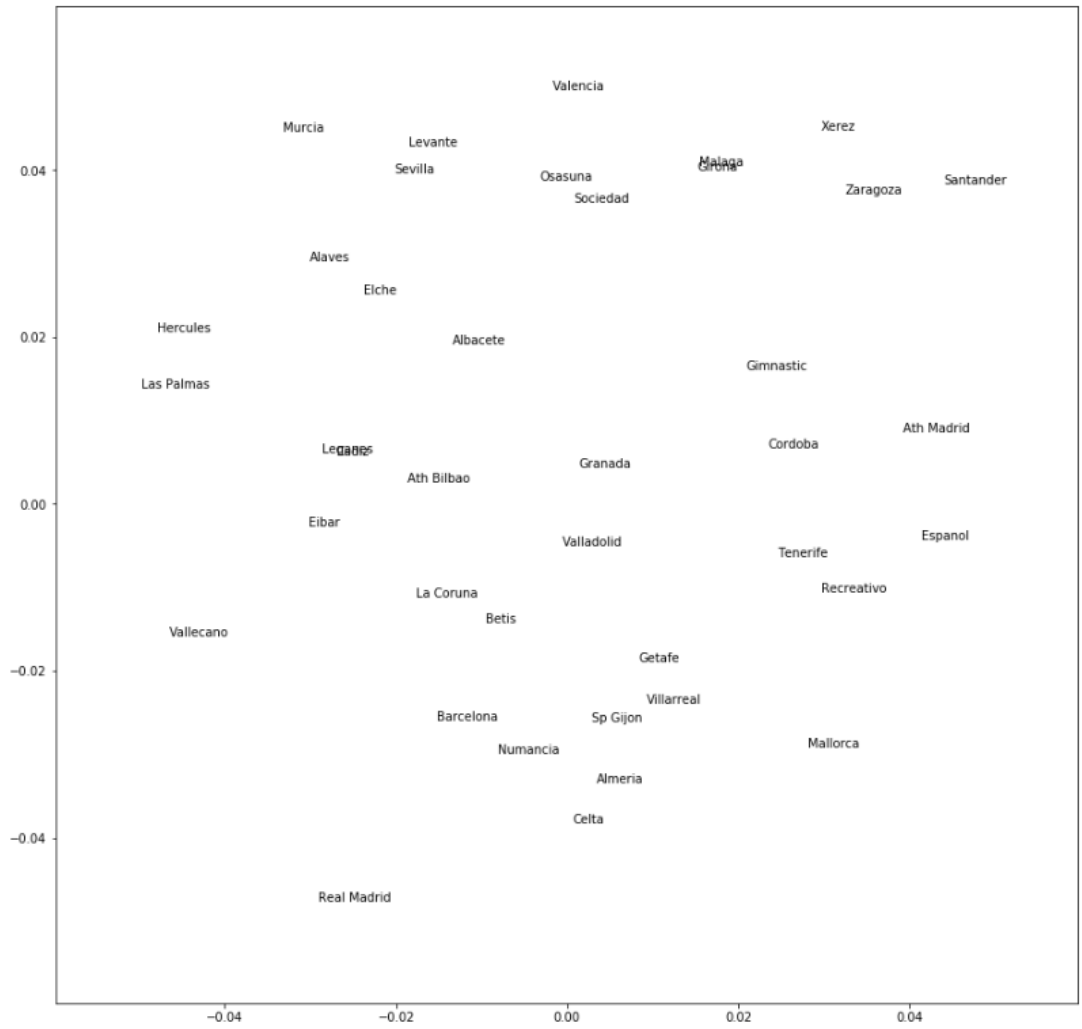

Figura 6.10: Representación 2D de los equipos.

```
array([[ 0,  0,  0, ...,  0,  0,  1],
       [ 0,  0,  0, ...,  0,  0,  1],
       [ 0,  0,  0, ...,  0,  0,  1],
       ...,
       [78, 75, 75, ..., 43, 43, 38],
       [90, 90, 87, ..., 43, 43, 38],
       [70, 67, 67, ..., 28, 27, 38]])
```

Figura 6.11: Datos sin escalar.

la tabla 6.1. Para poder hacer una comparación entre los 3 modelos se han usado los mismos cálculos para los tres, en el modelo estadístico se ha empleado la probabilidad más alta para predecir que va a pasar eso en el partido y compararla con lo que pasó y

```
array([[0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       ...,
       [0.80412371, 0.79787234, 0.82417582, ..., 0.48863636, 0.48863636,
        1.        ],
       [0.92783505, 0.95744681, 0.95604396, ..., 0.48863636, 0.48863636,
        1.        ],
       [0.72164948, 0.71276596, 0.73626374, ..., 0.31818182, 0.30681818,
        1.        ]])
```

Figura 6.12: Datos escalados en el rango [0,1].

llega a un total de 42.36 % de acierto. En los modelos de Deep Learning, se ha usado el método *predict* que nos proporciona Keras, para obtener las predicciones que hace sobre el conjunto de test. La probabilidad más alta ha sido usado para compararla con lo que pasó, es decir, si de un partido predice (devuelve un array con 3 probabilidades) 0.54 que gana el equipo local, 0.21 que empaten y 0.25 que gane el equipo visitante. Elegimos la más alta, como que el equipo local ganó, lo comparamos con lo que pasó y si ha acertado sumamos uno, en caso contrario, no hacemos nada. Al final, dividimos todos los aciertos por el número de partidos que hay en el conjunto de test y obtenemos el porcentaje de acierto. De esta manera se obtiene los porcentajes usando la misma métrica.

Modelo Estadístico	ML1	ML2
42.36 %	50.94 %	53.64 %

Tabla 6.1: Comparativa entre los distintos modelos.

7 Conclusiones

El objetivo principal del TFG era recopilar información sobre los partidos de la Liga Española y sus jugadores, este objetivo se ha cumplido exitosamente, se ha conseguido recopilar una gran cantidad de partidos, concretamente desde 1970. Además, hemos almacenado información sobre esos partidos, como goles, tarjetas, faltas, y una larga lista de características. Previo a toda la recopilación de la información se ha tenido que diseñar una BD y implementarla. Se han creado scripts de Python para poder crear la BD desde el inicio y mantener la información actualizada. Se decidió usar Python para el diseño y la implementación porque nos proporciona librerías y APIs con funciones ya hechas que nos facilitan el trabajo a la hora de crear bases de datos, redes neuronales, ficheros .csv, etc.

El segundo objetivo del proyecto era hacer pruebas con modelos estadísticos y con algoritmos de Deep Learning, se ha realizado evidenciando que es un problema muy complejo de resolver y dónde se pueden hacer cantidades ingentes de pruebas, ya sea cambiando la información que proporcionamos a los modelos, como cambiando los modelos, es decir, añadiendo capas y usando otro tipo de modelos de aprendizaje.

Algunos de los principales problema con los que nos hemos encontrado ha sido unificar toda la información de diversas fuentes, nos hemos encontrado con distintos nombres para el mismo equipo o información que se encontraba en un lugar pero en otro no. Otro problema con el que nos hemos encontrado ha sido el de decidir que factores hemos creído que influyen más en un partido y qué tipo de modelo usar, ya que por tiempo no se puede abarcar todos las posibles pruebas.

Como perspectivas de futuro, es un trabajo con gran escalabilidad ya que podemos llevarlo a otras ligas de fútbol y incluso a otros deportes, manteniendo la misma base. Las propuestas de futuro son seguir trabajando en el proyecto, probando nuevas combinaciones de datasets y nuevos modelos como redes Siamesas, Recurrentes, etc..

8 Anexo I

8.1. Base de Datos

Para el diseño del diagrama UML de la Base de Datos he usado una herramienta llamada draw.io que sirve para crear diagramas de todo tipo.

Para todo el almacenamiento de datos hemos usado XAMPP que es un paquete de software libre. Consiste principalmente en un sistema de gestión de bases de datos MySQL. [15]

Figura 8.1: Logo XAMPP

Una vez instalado XAMPP, lo ejecutamos y nos aparece un panel como 8.2 y tenemos que darle al boton de start de Apache y MySQL. Buscamos en el navegador la dirección de *localhost/phpmyadmin* y una vez dentro creamos la BD con el nombre y el tipo de alojamiento.

Para crear las tablas y relaciones entre ellas hemos usado scripts en python. La conexión a la BD se realiza de la siguiente manera, en los puntos suspensivos van los atributos de cada tabla, por simplificación los he reducido a puntos suspensivos.

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  #passwd="yourpassword",
  database="prueba"
)
```

```
9
10  mycursor = mydb.cursor()
11
12  mycursor.execute("CREATE TABLE Jugador (id MEDIUMINT NOT NULL
        AUTO_INCREMENT, ... , PRIMARY KEY (id)")
13
14  mycursor.execute("CREATE TABLE JugadorPartido (id MEDIUMINT NOT
        NULL AUTO_INCREMENT, ... , PRIMARY KEY (id)")
15
16  mycursor.execute("CREATE TABLE Equipo (id MEDIUMINT NOT NULL
        AUTO_INCREMENT, ... , PRIMARY KEY (id)")
17
18  mycursor.execute("CREATE TABLE EquipoPartido (id MEDIUMINT NOT
        NULL AUTO_INCREMENT, ... , PRIMARY KEY (id)")
19
20  mycursor.execute("CREATE TABLE Partido (id MEDIUMINT NOT NULL
        AUTO_INCREMENT, ... , PRIMARY KEY (id)")
```

Para rellenar la tabla hemos usado los siguientes datasets, comentados en el apartado de estructuración 4.

Para ir guardando la información de los .csv el orden de ejecución de los scripts es el que se muestra en la siguiente lista, para ejecutar tienes que irte al directorio y en la consola escribir *python "nombre.py"* donde nombre.py sera el nombre del archivo python.

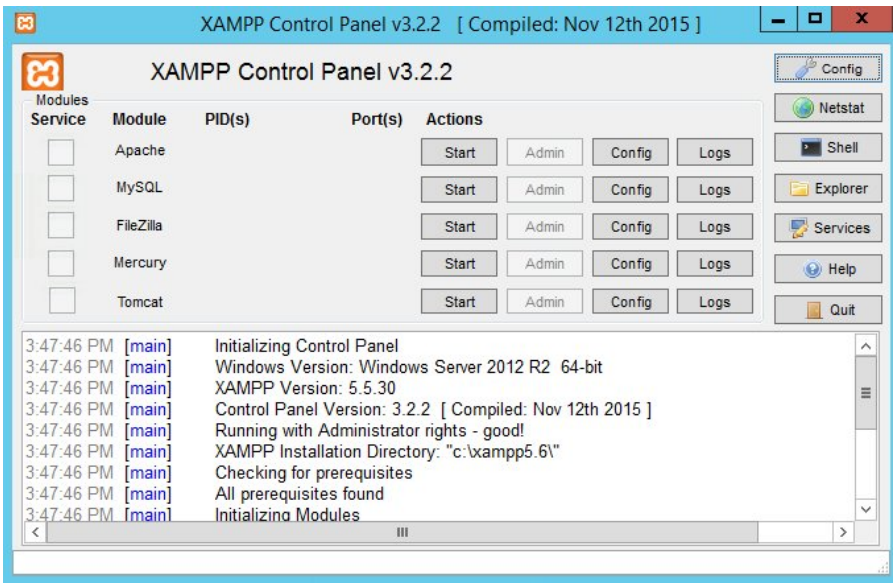

Figura 8.2: Panel XAMPP

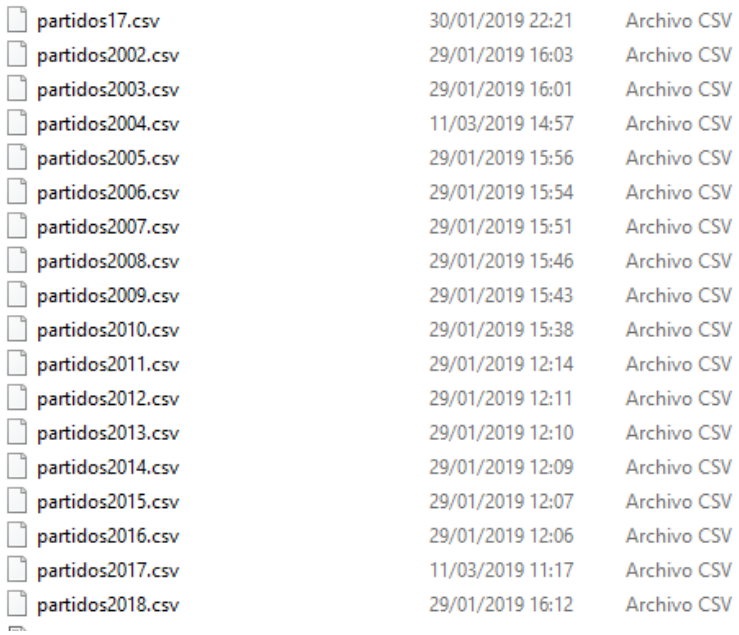

Figura 8.3: Lista de los archivos .csv

1. bd.py
2. jugadores.py
3. jugadores2k18.py
4. jugadorPartido.py
5. equipoPartido.py
6. equipoPartido2016.py
7. partido.py
8. partido2017.py
9. partidosfaltan.py

8.2. API keras y Colab

El API de keras nos proporciona implementaciones y métodos para crear Redes Neuronales, entrenarlas y que intenten predecir. [16] Las ANN son un modelo computacional que intenta emular el funcionamiento de las neuronas biológicas. Una red neuronal está compuesta por neuronas y por capas, la primera capa es la capa visible, a ella se le pasa los datos de entrada. Después puede tener capas ocultas que son capas intermedias interconectadas cada una con la anterior y con la siguiente. La última capa es la capa de salida, la cual proporciona una salida. Cada neurona tiene una función de activación, en nuestro caso *Relu*, la cual pone los valores entre 0 y 1.

```
from keras.layers import Input, Dense
from keras.models import Model

# This returns a tensor
inputs = Input(shape=(784,))

# a layer instance is callable on a tensor, and returns a tensor
x = Dense(64, activation='relu')(inputs)
x = Dense(64, activation='relu')(x)
predictions = Dense(10, activation='softmax')(x)

# This creates a model that includes
# the Input layer and three Dense layers
model = Model(inputs=inputs, outputs=predictions)
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
model.fit(data, labels)  # starts training
```

Figura 8.4: Ejemplo de Red Neuronal usando el API funcional de Keras

Vista de forma gráfica la red neuronal sería así pero con las neuronas descritas para cada capa. La siguiente red representaría una red con una capa de entrada con n neuronas, una capa oculta con m neuronas y una capa de salida con una neurona.

En el proceso de entrenamiento lo que realiza es una adaptación de los pesos de las conexiones para que cuando vea un nuevo dato, sepa clasificarlo correctamente. Si hiciera una clasificación correcta de un conjunto de entrenamiento ficticio con dos clases al 100 % quedaría como 8.6.

Google Colaboratory es un cuaderno de python que nos proporciona Google, en el que podemos usar GPU, que nos viene muy bien a la hora de entrenar una red ya que se necesita capacidad de cómputo y con la GPU nos proporciona paralelización de las operaciones lo que hace que se ejecute mucho más rápido. Lo que hemos hecho es subir también el dataset a Google Drive y cargarlo desde el Colab, de este modo cada vez que cerremos no perdemos los datos.

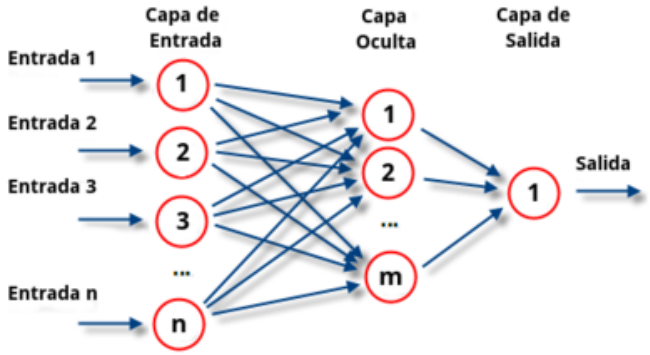

Figura 8.5: Ejemplo de Red Neuronal vista de forma gráfica

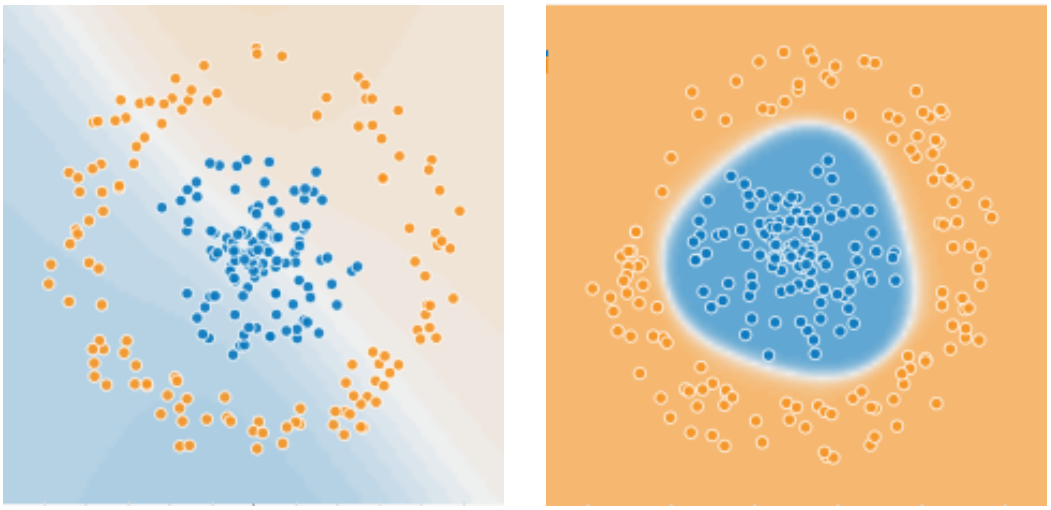

Figura 8.6: Antes (izquierda) y después del entrenamiento (derecha).

Bibliografía

[1] Fútbol deporte más popular. https://www.bloomberg.com/news/articles/2018-06-12/soccer-is-the-world-s-most-popular-sport-and-still-growing. Accessed: 2019-01-12.

[2] Estadisticas de fútbol. https://www.statista.com/topics/1595/soccer/. Accessed: 2019-01-12.

[3] Página con información sobre partidos. https://www.soccerbase.com. Accessed: 2019-02-20.

[4] Página con información sobre partidos. https://footballresults.org/league.php?table=1&league=PrimeraDiv. Accessed: 2019-02-20.

[5] Página con información sobre partidos. https://rapidapi.com/api-sports/api/api-football. Accessed: 2019-02-20.

[6] Qué tienen las casas de apuestas para predecir. https://www.sciencedirect.com/science/article/pii/S0167268109000833. Accessed: 2019-02-20.

[7] Microsoft bing predijo mundial 2014. https://searchengineland.com/bing-predicts-world-cup-193846. Accessed: 2019-02-20.

[8] Predicción usando redes bayesianas. https://www.sciencedirect.com/science/article/abs/pii/S0950705106000724. Accessed: 2019-02-20.

[9] Besoccer. https://itunes.apple.com/es/app/resultados-de-f%C3%BAtbol-besoccer/id550928207?mt=8. Accessed: 2019-02-20.

[10] Estudio con distribucción de poisson. https://towardsdatascience.com/predicting-premier-league-standings-putting-that-math-to-some-use-e8de64938d7. Accessed: 2019-01-10.

[11] Distribución de poisson. https://es.wikipedia.org/wiki/Distribuci%C3%B3n_de_Poisson. Accessed: 2019-02-20.

[12] Interpolar datos en una función. https://apps.dtic.mil/dtic/tr/fulltext/u2/a240394.pdf. Accessed: 2019-01-27.

[13] Series de fourier. https://es.wikipedia.org/wiki/Serie_de_Fourier. Accessed: 2019-02-13.

[14] Embeddings. https://towardsdatascience.com/node2vec-embeddings-for-graph-data-32a866340fef. Accessed: 2019-02-14.

[15] Xampp. https://www.apachefriends.org/es/index.html. Accessed: 2019-01-12.

[16] Keras. https://keras.io/. Accessed: 2019-02-14.